Universidad Autónoma de Madrid

Escuela Politécnica Superior

Grado en Ingeniería Informática

TRABAJO DE FIN DE GRADO

APLICACIÓN PARA EL DESCUBRIMIENTO E INTEGRACIÓN DE DATOS CON PROTOCOLO ONVIF

Candelaria Orellana Peró
Tutor: Eduardo Cermeño Mediavilla
Ponente: Juan Alberto Sigüenza Pizarro

Mayo 2020

APLICACIÓN PARA EL DESCUBRIMIENTO E INTEGRACIÓN DE DATOS CON PROTOCOLO ONVIF

AUTOR: Candelaria Orellana Peró
TUTOR: Eduardo Cermeño Mediavilla
PONENTE: Juan Alberto Sigüenza Pizarro

Dpto. de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid

Mayo 2020

Agradecimientos

A mis padres, por tomar la decisión de mudarnos a España y tener así todas las oportunidades que mis hermanos y yo hemos tenido.

A mi amiga Sofía, por ser un apoyo incondicional a pesar de la distancia que nos separa.

A mis amigos de la carrera, que juntos pudimos superar esta etapa llena de desafíos con sus altos y bajos.

A mis amigas de hockey, por apoyarme cuando no estaba teniendo un buen día y poder contar con ellas en cualquier momento.

Resumen

Resumen — En este Trabajo de Fin de Grado se ha implementado un Bridge ONVIF. Debido al uso de HTTP para la integración de datos y para la comunicación con un servidor resulta interesante disponer de un Bridge que pueda conectarse con sistemas ONVIF que son aquellos que cuentan con dispositivos compatibles con el estándar ONVIF. El Bridge consta del desarrollo de un módulo que permite la integración de una notificación HTTP en un servidor ONVIF. Esto permitirá la comunicación entre un sistema externo y un servidor ONVIF. Para ello se implementarán cuatro módulos: un sistema externo encargado de enviar dichas notificaciones, un servidor web que junto a un cliente ONVIF serán el Bridge ONVIF y un servidor ONVIF que recibirá dichas notificaciones. El servidor web será el encargado de reenviar las notificaciones y el cliente se comportará como un buscador ya que tendrá que buscar los servidores ONVIF en red.

Por otro lado, se han implementado algunos aspectos del estándar ONVIF, como el descubrimiento de dispositivos en red, el envío de la url para la reproducción de vídeo en vivo y la suscripción de eventos generados por un dispositivo. ONVIF es el principal estándar utilizado para la comunicación de dispositivos de seguridad. El objetivo de este es permitir que los dispositivos que lo implementen puedan compartir una interfaz y tener la posibilidad de intercambiar información, obteniendo una *interoperabilidad* entre los productos con independencia del fabricante. Este estándar será utilizado en el caso del Servidor ONVIF que se encargará de reproducir la imagen en vivo de un dispositivo previamente seleccionado.

Palabras clave — ONVIF, descubrimiento, suscripción, evento, interoperabilidad, Bridge, notificación, HTTP

Abstract

Abstract — In this Final Degree Project, an ONVIF Bridge has been implemented. Due to the use of HTTP for data integration and for communication with a server, it is interesting to have a Bridge that can connect to ONVIF systems, which are those that have devices compatible with the ONVIF standard. The Bridge consists of the development of a module that allows the integration of an HTTP notification in an ONVIF server. This will allow communication between an external system and an ONVIF server. For this, four modules will be implemented: an external system in charge of sending said notifications, a web server that together with an ONVIF client will be the Bridge ONVIF and an ONVIF server that will receive said notifications. The web server will be in charge of forwarding the notifications and the client will behave like a search engine since it will have to search the ONVIF servers on the network.

On the other hand, some aspects of the ONVIF standard have been implemented, such as the discovery of networked devices, the sending of the url for the reproduction of live video and the subscription of events generated by a device. ONVIF is the main standard used for communication of security devices. The objective of this is to allow the devices that implement it to share an interface and have the possibility of exchanging information, obtaining a *interoperability* between the products regardless of the manufacturer. This standard will be used in the case of the ONVIF Server that will be in charge of reproducing the live image of a previously selected device.

Key words — ONVIF, discovery, subscription, event, interoperability, Bridge, notification, HTTP

Índice general

Índice de figuras

1 Introducción

A continuación, se explicará la idea general del trabajo describiendo la motivación de su realización, los objetivos que se van a cumplir y una breve estructura del mismo.

1.1 Motivación y Objetivos

La seguridad física tardó cierto tiempo en poner en valor las ventajas de la tecnología IP. *"La seguridad física se utiliza para proteger el sistema informático utilizando barreras físicas y mecanismos de control de forma análoga"* [1]. El uso de la tecnología IP en el área de la seguridad física supuso una gran ventaja, ya que por un lado esta unión permite garantizar la seguridad del acceso a los servidores. Por otro lado, en los últimos años los dispositivos creados para ser utilizados en este rubro permiten la conectividad a través de la red IP.

Existe un problema en el desarrollo de estos dispositivos, y es que cada uno de ellos es diseñado para que la comunicación e integración sea compatible entre productos del mismo fabricante. Esto es una desventaja, puesto que los clientes se ven obligados a utilizar dispositivos del mismo proveedor en un mismo sistema. Es por ello que la estandarización y la interoperabilidad cobran importancia en el avance de la seguridad física. Open Network Video Interface Forum (ONVIF) nace de la necesidad de tener un estándar abierto que permita la interoperabilidad entre dispositivos de distintos proveedores, permitiendo así que los clientes puedan, por ejemplo, contar con un software de control de acceso de un fabricante y en un futuro adquirir uno de otro proveedor sin tener que realizar modificaciones en su sistema.

En este TFG se desarrollará un Bridge ONVIF, como su nombre lo indica, es un "puente" que une dos redes conectadas. Para la implementación serán desarrollados cuatro módulos, un Servidor ONVIF que podrá ser instalado en cualquier máquina o dispositivo conectado a una red de forma que cuando se descubra algún producto compatible con ONVIF en la misma red, se podrá recibir información auténtica. Un módulo Cliente ONVIF que se encargará de actuar como un buscador. Una aplicación con interfaz privada que será la que enviará notificaciones HTTP al Bridge. Por último, se desarrollará un Servidor Web, que recibirá las notificaciones enviadas por

la aplicación externa y las reenviará al servidor ONVIF. Por tanto, el Bridge ONVIF permitirá la integración HTTP en un sistema ONVIF. El Bridge contará con una dirección y puerto fijo, de esta forma podrá ser accedido por cualquier sistema externo que se encuentre en la misma red.

En el mercado hay miles de sistemas que se comunican utilizando llamadas HTTP y las empresas en el rubro de la seguridad física las usan con mucha frecuencia. El bridge que se desarrollará en este proyecto puede ser utilizado por las empresas que deseen integrar la seguridad física e informática, ya que se podrá almacenar la información obtenida por una aplicación inteligente en un sistema ONVIF como por ejemplo en un VMS. Un caso en el que podría ser utilizado es, por ejemplo, la entrada de una oficina es controlada mediante una cámara de seguridad que tiene la capacidad de reconocer a un empleado ya que cuenta con reconocimiento facial. Suponiendo que la cámara no es compatible con ONVIF, cada vez que reconozca a un empleado podría enviar la información recogida mediante notificaciones HTTP al Bridge ONVIF, ya que la dirección del VMS no es pública. Este será el encargado de reenviárselas al VMS y se podría manipular la información obtenida para conceder el acceso a los empleados. Otro caso donde podría ser de utilidad, es en una empresa que cuenta con un número variable de grabadores, el buscador será el encargado de buscar las direcciones de estos y serán los que recibirán las notificaciones recibidas.

Aquellas empresas o edificios que cuenten con un sistema de vigilancia y/o de control de acceso con dispositivos compatibles con ONVIF podrían sacarle partido al Bridge ONVIF, puesto que podrán almacenar cualquier información recibida tanto por aplicaciones como por dispositivos IoT, como los sensores conectados a la red que se utilizan en plantas de producción que analizan datos y pueden generar alarmas o mensajes.

1.2 Estructura del documento

Esta memoria se ha estructurado siguiendo las distintas fases establecidas en el diseño:

- En el *Capítulo 1* se ha realizado una breve introducción del proyecto, donde se establece la motivación y los objetivos del mismo.
- En el *Capítulo 2* se realiza un análisis de quienes son los que utilizan ONVIF y con qué fin. Además, se explicará brevemente la estructura del estándar ONVIF, el concepto de los perfiles y cuáles son los dispositivos, que a día de hoy, son compatibles con el protocolo.
- En el *Capítulo 3* se expone la fase de Diseño donde se detallan los componentes software que se desarrollarán, las tecnologías utilizadas y se analizan en más detalle algunas de las características de ONVIF. También se hará tanto un análisis de los requisitos que conforman el proyecto como el patrón de diseño que se utilizará.
- En el *Capítulo 4* se expone la fase de Desarrollo donde se detalla cómo se implementaron cada uno de los componentes indicados en la fase de Diseño y las herramientas que fueron utilizadas.
- En el *Capítulo 5* se exponen y se describen los resultados de las pruebas realizadas durante y tras finalizar la fase de desarrollo.
- En el *Capítulo 6* se realiza una valoración del objetivo y se exponen cuáles son las principales cualidades del software creado. Además, se plantean las mejoras para el trabajo futuro.

2

Estado del arte

Previo a la fase de Diseño del proyecto es necesario comprender y familiarizarse con el protocolo ONVIF. Es por ello que en esta sección, en primer lugar, se expondrán algunas de las características principales. En segundo lugar, se analizan quienes son los que utilizan ONVIF. Por último, se explicará como es la estructura del estándar y cuáles son los dispositivos que lo utilizan.

2.1 Orígenes y características

En 2008, tres grandes compañías como *Axis Communications, Bosch Security Systems* y *Sony Corporation* crearon la organización sin ánimo de lucro Open Network Video Interface Forum (ONVIF), *"cuyo objetivo principal es proporcionar y promover interfaces estandarizadas para la interoperabilidad efectiva de los productos de seguridad física basados en IP"*. [2]

Una de las características más importantes de ONVIF es la interoperabilidad que se facilita entre los distintos dispositivos IP, sin tener en cuenta el fabricante. Esto permite que los productos de seguridad física basados en IP tengan una única interfaz de comunicación estandarizada, conseguida gracias a los servicios web y al streaming en vivo. Otra peculiaridad que resaltan los beneficios de implementar ONVIF es el hecho de que está abierta a todas las empresas y organizaciones. Esto potencia la estandarización de la comunicación y la interoperabilidad. Para garantizar la interoperabilidad, todos los dispositivos ONVIF utilizan el patrón de la Organización para la Interoperabilidad de Servicios Web (WS-I).

Es una organización compuesta por empresas, miembros, que se encuentran en el área de la seguridad IP. Actualmente ONVIF cuenta con una base sólida de miembros distribuida en los seis continentes, en la Figura 2.1 podemos ver reflejados que los países asiáticos son los miembros mayoritarios. En ONVIF, las decisiones se deciden entre los miembros, es decir, es gobernada por ellos. Dentro de estos, se encuentran los líderes en cámaras, sistemas de gestión de video y desarrollo de acceso de control.

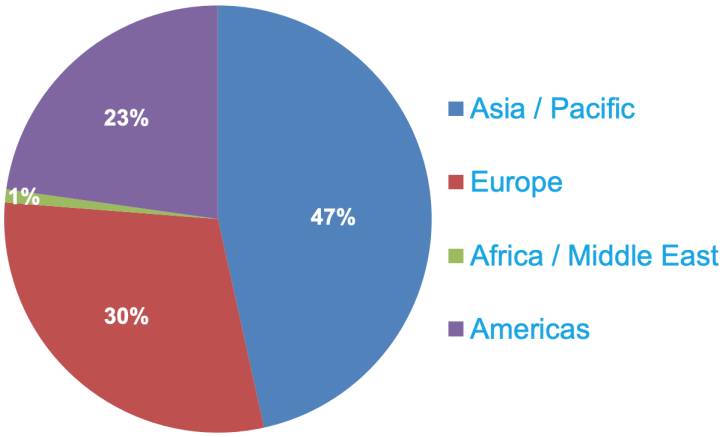

Figura 2.1: Distribución geográfica de los miembros.
Fuente: onvif.org

Existen distintos tipos de miembros y cada uno de ellos tienen distintas contribuciones [3]:

- **Plenos y Contribuyentes:** pueden incluir activamente en el desarrollo del estándar al participar en el trabajo de la organización.
  - **Empresas con membresía plena:** *Canon, Cisco, Huawei*, entre otros.
  - **Empresas con membresía contribuyente:** *Microsoft, Flir, AxxonSoft*, entre otros.
- **Usuario:** está abierto a las organizaciones que deseen utilizar la especificación de la interfaz de red y tengan acceso a propuestas de especificación, pero no quieren participar en ningún trabajo de la organización. Empresas con esta membresía: *Novus, Fujifilm, Mitsubishi Electric*, entre otros.
- **Observador:** está abierto a organizaciones que no desean participar en ningún trabajo dentro de la organización ONVIF, pero se les otorga ciertos beneficios limitados, como el derecho a acceder a las herramientas de prueba de especificación de interfaz. Sin embargo, los miembros observadores no pueden presentar, reclamar, comercializar o promocionar ningún producto de hardware o aplicación de software u otro dispositivo para ser clasificado como un producto de red compatible. Las compañías que son fabricantes de dispositivos o vendedores de software de clientes no pueden unirse a este tipo de membresía. Empresas con esta membresía: *AKVA Group, Dynacom Communications, Edimax*, entre otras.

2.2 Dispositivos y Estructura

En esta sección se comentarán alguno de los dispositivos que utilizan ONVIF y la estructura del estándar. En primer lugar, se procederá a explicar la estructura, en concreto se explicará los perfiles con los que cuenta. Por último, se detallarán algunos de los dispositivos que son utilizados con el protocolo ONVIF y se expondrán algunos casos donde son empleados. Hay muchos dispositivos que utilizan el protocolo, pero en este proyecto se creará un Bridge ONVIF.

2.2.1. Estructura

ONVIF cuenta con muchas especificaciones que se agrupan en los llamados perfiles. *"Un perfil ONVIF data de características fijas y permite identificar qué productos son compatibles entre sí"* [4]. Un producto ONVIF tiene que por lo menos cumplir un perfil. Las características establecidas en cada perfil deben ser cumplidas tanto por el dispositivo como por el cliente. Las características establecidas de un perfil pueden ser obligatorias o condicionales. Las condicionales deben cumplirse solo si son relevantes para los dispositivos de dicho perfil. En cambio, como su nombre lo indica, las obligatorias deben ser cumplidas para pertenecer a ese perfil.

Features		Profiles											
		G		Q		S		T		C		A	
		Device	Client	Device	Client	Device	Client	Device	Client	Device	Client	Device	Client
General													
System Settings		M	C	M	C	M	C	M	C	M	C	M	C
User Authentication	WS-Username Token					M	M						
	Digest Authentication	M	M	M	M	O	M	M	M	M	M	M	M
User Handling		M	C	M	M	M	C	M	C	M	C	M	M
Query Services and Capabilities		M	M	M	M	M	M	M	M	M	M	M	M
Device Discovery		M	C	M	M	M	C	M	M	M	C	M	M
Default Access Policy				M									
Network Configuration		M	C	M	C	M	C	M	M	M	M	M	C
Zero configuration				C	C	C	C						
Firmware Upgrade				C	C								
Backup and Restore				C	C								
TLS Configuration				C	C								
IP Address Filtering						C	C					C	C
NTP						C	C	C	C			M	C
Automatic IP Assignment				C	C								
Media Profile Configuration						M	C	M	C				
Media Transport	RTP/UDP	M	M			M	M[1]	M	M[1]				
	RTP/RTSP/HTTP/TCP	M	M			M		M					
	RTP/RTSP/HTTPS/TCP	C	C					C	C				
	RTP/RTSP/TCP/WebSocket							C	O				
	RTP/UDP Multicast					C	C	M	C				

Figura 2.2: Características generales de los perfiles ONVIF
Fuente: ONVIF Profile Feature Overview v2.3

En la actualidad, ONVIF cuenta con seis distintos perfiles [5]:

- **Perfil S.** En 2011, tras tres años de la creación de la organización, se creó el primer perfil que permite la transmisión de vídeo básica. El fin de un dispositivo compatible con este perfil es transmitir datos de vídeos por medio de una red IP a un cliente de perfil S. Un cliente de este perfil debería ser capaz de controlar, configurar y solicitar la transmisión de vídeo de un dispositivo.
- **Perfil C.** Luego de dos años, en 2013, se creó el segundo perfil para el control de puerta y la gestión de eventos. Este perfil permite la conectividad entre dispositivos de vídeo y de acceso. Tanto el cliente como el dispositivo pueden acceder a la información obtenida, como controlar el acceso a la puerta y gestionar los eventos y las alarmas.
- **Perfil G.** Tras la creación del perfil C, la organización creó este perfil para la grabación y el almacenamiento local. Gracias a este perfil ya no es necesario enviar el vídeo a un

almacenamiento externo, como un servidor central, ya que se puede almacenar directamente en el dispositivo utilizando una memoria SD o de forma local utilizando un dispositivo de almacenamiento conectado directamente. En sus especificaciones se incluye la posibilidad de grabar audio y metadata, que podrán ser recibidas sólo si el cliente las soporta.

- **Perfil Q.** Este perfil permite tanto la configuración, descubrimiento y control de dispositivos compatibles como la instalación rápida. Además, cuenta con las especificaciones para que la comunicación entre dispositivo y cliente, ambos pertenecientes al perfil, sean seguras. Para ello utiliza el protocolo Seguridad de la Capa de Transporte (TLS) que encripta las comunicaciones que se realizan mediante la red IP.

- **Perfil A.** Debido a la demanda de aumentar la funcionalidad de los sistemas de control de acceso, la organización ONVIF creó el perfil A, concediendo a los dispositivos la posibilidad de configuración las reglas de acceso, credenciales y horarios además de recuperar la información, estados y eventos. El cliente podrá proporcionar las configuraciones para las credenciales, los horarios y el control de acceso.

- **Perfil T.** Se creó este perfil para dar soporte al streaming de vídeo avanzado. Este permite audio bidireccional, configuraciones de imagen, compresión de vídeo H.264/H.265, eventos de alarma de movimiento y manipulación y transmisión de metadatos. El dispositivo debe contar con la posibilidad de visualizar en pantalla el streaming. Mientras que el cliente también tendrá la oportunidad de controlar el control PTZ de la cámara. Otra peculiaridad de este nuevo perfil es la incorporación del protocolo HTTPS para la transferencia de datos, siempre y cuando el dispositivo y el cliente lo soporten.

  Puesto que el perfil S también es un perfil creado para la transmisión de vídeo, la creación de este perfil no implica la sustitución del perfil S, ya que cada uno tiene sus propias áreas y se pueden combinar.

2.2.2. Dispositivos que utilizan ONVIF

En el comienzo se puede comprobar que los productos que incluía este protocolo eran los relacionados con el vídeo, como las cámaras, analíticas, grabadores de vídeo y codificadores de vídeo. Pero, tras la creación del perfil A, se decidió incluir los productos relacionados con el control de acceso. En esta sección se detallarán algunos de dichos productos.

En la página oficial de la organización [6], se puede comprobar si un dispositivo pertenece o no a ONVIF. En caso de no pertenecer, se puede informar de su compatibilidad con el protocolo rellenando un formulario.

2.2.2.1. Cámaras de Red

Una cámara de red o también conocida como cámara IP [7] es un dispositivo que permite, mediante una red IP, transmitir señales de vídeo, y dependiendo de la cámara también es posible la transmisión de audio, a otro dispositivo como por ejemplo un VMS, NVR, ordenador o mismo un SmartPhone. Gracias a estos dispositivos, los usuarios cuentan con la posibilidad de almacenar, gestionar y visualizar dichas transmisiones en tiempo real. Esto se consigue mediante protocolos de streaming en vivo, un servidor web y una dirección IP determinada para la cámara. Además, el usuario puede utilizar un navegador web o una aplicación para poder visualizar la salida de vídeo desde cualquier ubicación, tanto remota como local.

Las cámaras de red comúnmente contienen un servidor web integrado que se puede acceder y controlar desde cualquier red IP como WAN, LAN o Internet. Estas combinan las funcionalidades de una cámara con algunas de las funcionalidades de un ordenador, no requieren estar conectadas a un ordenador para poder operar. Tienen su propia dirección IP y se pueden conectar mediante un cable *Ethernet* o por conexión inalámbrica, pero requieren mantenimiento. [8]

Este dispositivo puede ser compatible con los *perfiles S, G, Q y T* puesto que estos son los perfiles relacionados con la transmisión de vídeo. Las cámaras IP son generalmente empleadas para la vigilancia de alguna entrada o, por ejemplo, para la seguridad de un aeropuerto. En ambos casos, si la cámara lo dispone se puede grabar la imagen en vivo en la memoria SD.

2.2.2.2. Grabador de video digital (DVR)

Las grabadoras de vídeo dependen de discos duros para poder almacenar los datos digitales. Pueden grabar un flujo de entrada de vídeo desde una variedad de fuentes como cámaras web, cámara de video, entre otras grabando los datos de vídeo y almacenando los datos en un disco duro. Además de contar con las funcionalidades de grabación, reproducción, avance rápido, rebobinar y pausar, los DVR cuentan con la habilidad de poder saltar a cualquier parte de la grabación sin tener que rebobinar o avanzar rápidamente el stream. [9]

Si bien los DVR tienen muchas ventajas como la de ser sin cinta, recuperación de datos más rápida y una mayor calidad de imagen, no pueden admitir varias cámaras en un solo sistema en comparación con otros ni tampoco pueden procesar tantos cuadros por segundo. [10]

2.2.2.3. Grabador de Vídeo de Red (NVR)

Un NVR es un dispositivo que cuenta con un software de administración preinstalado en él. Es un sistema que incluye la computadora, el software, el almacenamiento y dependiendo del grabador incluye también múltiples puertos de alimentación a través de Ethernet (POE). Algunas de las funcionalidades de estos dispositivos son la de grabar, almacenar, analizar y reproducir vídeos de red. Los NVR rara vez permiten modificarse para agregar algo fuera de su especificación original. [11]

Una de las diferencias entre los DVR y NVR, es que los DVR comprimen las señales de vídeo de un dispositivo análogo digitalmente. En cambio, en los NVR se graban las señales de vídeo que fueron previamente codificadas por las cámaras. Es por ello que las entradas y salidas de los NVR son datos IP y no requieren ninguna conexión de vídeo.

La mayor ventaja de estos dispositivos es que no requieren estar en un lugar determinado, pueden estar instalados en cualquier lugar de la red como cerca de un grupo de cámaras, protegidos en un entorno o por ejemplo en un centro de control. Otra ventaja es que tienen la capacidad de visualizar y grabar imágenes al mismo tiempo. Además, permiten que los usuarios puedan observar estas imágenes simultáneamente y desde distintos puntos de la red independientemente, es decir, no afectará a los demás usuarios. [12]

Si una cámara IP es configurada para transmitir vídeo a una grabadora, ambos deben ser compatibles con el *perfil S*, que es el más básico y permite a las cámaras y NVR solicitar y controlar la transmisión de vídeo. El fin de un dispositivo compatible con este perfil es transmitir datos de vídeo por medio de una red IP a un cliente S, por ejemplo un NVR.

2.2.2.4. Video Management System (VMS)

Generalmente en un sistema que cuenta con cámaras IP, las cámaras apuntan a un sistema de administración de vídeo (VMS) centralizado, que se utiliza para visualizar, reproducir y grabar el vídeo [13]. El VMS es un elemento central de un sistema de seguridad que principalmente consiste de cámaras. Existe la posibilidad de que en un VMS se integren dispositivos diferentes a las cámaras, como por ejemplo los dispositivos de control de acceso o altavoces. Se utiliza, en general, para controlar todos los dispositivos conectados a él y también para recopilar datos de las diferentes fuentes. [14]

Algunos VMS tienen la capacidad de detectar movimiento permitiendo comenzar la grabación en cuanto se haya detectado un movimiento, de esta forma se ahorra espacio de almacenamiento. Otra característica, es que con este dispositivo existe la posibilidad de poder visualizar la imagen de múltiples cámaras a la vez. Otra cosa importante que hacen las soluciones VMS es revisar y analizar el vídeo grabado para luego devolver información útil.

Si vamos más allá en la tecnología avanzada actualmente presente en algunas soluciones VMS, se siguen encontrando más funciones, como el reconocimiento de matrículas. Esta tecnología permite registrar la matrícula de un coche para luego compararlo con una base de datos. Esta funcionalidad es utilizada por los departamentos de policía para detectar infracciones por exceso de velocidad, y también lo utilizan algunos aparcamientos para permitir la entrada y salida sin necesitar un ticket. [15]

Gracias al estándar ONVIF, este dispositivo puede consistir en cámaras u otros dispositivos de distintos fabricantes siempre y cuando cumplan el mismo perfil. Si se desea trabajar con un sistema que integra VMS, se debe utilizar el perfil G que permite las funciones de grabar, buscar y reproducir. El dispositivo que cumple con este perfil, como una cámara de red, es aquel que puede grabar datos de vídeo y almacenarlos en un su memoria SD, si la dispone. El VMS podría recuperar los datos de la SD y volver a reproducir la grabación, por ejemplo.

3 Diseño

3.1 Introducción

En esta sección se comienza detallando algunas de las características principales del protocolo, como su arquitectura basada en servicios web, y se determinará cuales son los perfiles utilizados en este proyecto. Una vez que estamos familiarizados con el concepto del protocolo ONVIF se procederá con el análisis de los requisitos que serán las condiciones y capacidades necesarias para alcanzar el objetivo. Además, se definirá el patrón de diseño utilizado en este proyecto.

Por último, se pasará a explicar la estructura de los módulos desarrollados en el trabajo. Cabe destacar que no se hará una implementación completa del estándar, se implementarán el descubrimiento de dispositivos en red, el envío de la url de streaming y la suscripción de eventos junto con el envío de notificaciones.

3.2 Protocolo ONVIF

En el capítulo anterior se han explicado los diferentes perfiles que se han creado estos años. Cabe destacar que en este proyecto se van a utilizar los perfiles G, Q, S y T puesto que son los que cuentan con especificaciones para los sistemas de vídeo.

3.2.1. Arquitectura y servicios

La arquitectura de ONVIF está basada en los servicios web. Un Servicio Web (WS) es una aplicación o fuente de datos que es accesible mediante los protocolos web estándares HTTP o HTTPS. Al contrario de las aplicaciones web, los servicios web se comunican con programas, en lugar de comunicarse con usuarios. [16]

A diferencia de otros servicios, los WS utilizan Web Services Description Language (WSDL)

para formular lo que pueden hacer, como llamarlo y lo que son. Comúnmente para enviar peticiones con formato XML, los WS utilizan el protocolo SOAP.

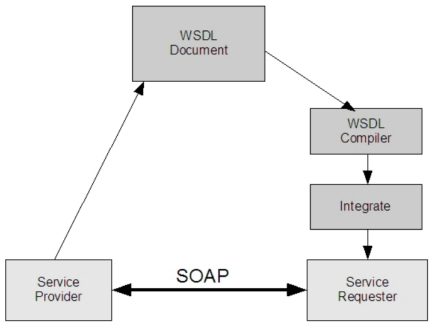

Figura 3.1: Arquitectura de los WS
Fuente: ONVIF Core Specifications - Ver. 19.12

En la Figura 3.1 podemos observar cómo es la arquitectura de los WS. El *Service Provider*, que se trata del dispositivo compatible con ONVIF, implementa el o los servicios utilizando WSDL basado en XML. Una vez ya creados los servicios, el cliente *Service Resquester* puede utilizarlos como base para el servicio de integración. Gracias a los compiladores de WSDL, el cliente puede integrar los WS dentro de una aplicación. La comunicación entre el cliente y el dispositivo, se realiza empleando el protocolo SOAP, que codifica la petición en un WS y la respuesta antes de enviar el mensaje por la red.

3.2.2. Beneficios

Con el transcurso de los años la popularidad del estándar ONVIF ha ido aumentando como también los beneficios para cada parte involucrada. Los usuarios finales que optan por utilizar ONVIF obtienen una mayor posibilidad a la hora de elegir los productos. Tendrán la opción de elegir un VMS de un proveedor y adquirir otros productos de otro fabricante sin perjudicar su sistema, puesto que existe la interoperabilidad entre los dispositivos. La flexibilidad en la elección de los dispositivos beneficia a los usuarios finales también en disminuir el coste de integración, ya que podrán elegir la combinación de productos que menor coste suponga. Además, contar con un sistema donde sus dispositivos son compatibles con ONVIF implica que su futuro está asegurado, es decir, a la hora de reemplazar o actualizar alguna parte del sistema no habrá riesgo alguno. Otro aspecto positivo, es que podrán emplear interfaces ONVIF y sus propias interfaces para distintas funcionalidades al mismo tiempo [17] [18].

Los integradores cuentan con unos beneficios semejantes a los usuarios finales. Cuentan con la posibilidad de crear sistemas flexibles y rentables tras la elección de productos compatibles con ONVIF. Asimismo, tienen la opción de elegir distintos proveedores, ya que la interoperabilidad está asegurada y gracias a ella, se facilita la integración de dichos productos [19].

Por último, cabe destacar los beneficios que obtienen las empresas de hardware y software. Para las empresas que se encargan de desarrollar el software, se reduce el tiempo invertido en el desarrollo de interfaces específicas para la funcionalidad básica de cada marca y, de esta forma, podrán centrarse, por ejemplo, en el desarrollo de soluciones innovadoras.

A las empresas hardware les interesa que su producto sea compatible con ONVIF, puesto

que les permite acceder a los proyectos suponiendo un aumento en las ventas. Además, implica ser aceptado en el mercado. Es importante destacar que para que un producto sea compatible con ONVIF, su proveedor deber ser miembro y además, dichos producto debe cumplir con por lo menos una de las especificaciones de un algún perfil.

3.3 Análisis de Requisitos

El análisis de requisitos es la fase donde se específica completamente el comportamiento que se espera del software a desarrollar. Esta fase es tanto importante para el cliente como para el desarrollador, ya que el impacto de cometer errores puede resultar en no satisfacer las necesidades del cliente.

3.3.1. Requisitos Funcionales

- **Subsistema de Envío**
  - **RF01:** el usuario podrá introducir una dirección IP válida a donde se enviarán las notificaciones.
  - **RF02:** la aplicación debe permitir el envío de notificaciones HTTP a una dirección.
- **Subsistema de Reenvío**
  - **RF03:** el Servidor Web debe disponer de una dirección y puerto fijo.
  - **RF04:** la aplicación debe ser capaz de reenviar las notificaciones recibidas al Servidor ONVIF.
- **Subsistema de Buscador**
  - **RF05:** la aplicación permitirá buscar en la red los servidores ONVIF disponibles.
  - **RF06:** la aplicación debe permitir la conexión entre el Servidor Web y el Cliente ONVIF.
  - **RF07:** la aplicación debe permitir al Cliente ONVIF seleccionar una dirección de un Servidor ONVIF.
  - **RF08:** la aplicación debe permitir al Cliente ONVIF enviar la dirección, previamente seleccionada, al Servidor Web.
- **Subsistema de Recepción**
  - **RF09:** la aplicación debe permitir recibir cualquier notificación HTTP enviada.
  - **RF10:** la aplicación permitirá la recepción de la dirección IP enviada por el buscador.
  - **RF11:** la aplicación almacenará las notificaciones recibidas.

3.3.2. Requisitos No Funcionales

- **RNF01:** es necesario que los componentes de la aplicación estén conectados en la misma red.
- **RF02:** es necesario que el Servidor ONVIF esté conectado y activo antes de que se establezca una conexión entre el Cliente ONVIF se comunique con la aplicación.

3.4 Patrón de diseño

El patrón de diseño que se utilizará en el desarrollo del proyecto es el Modelo-Vista-Controlador (MVC) [20]. Es un patrón que está estructurado en tres componentes independientes, el modelo, la vista y el controlador, donde se destaca la separación entre la lógica de negocio y la interacción con la vista.

En el modelo se definen los datos de la aplicación, es decir, los datos con los que trabajará el usuario. La vista establece cómo se reflejan los datos de la aplicación, es decir, son las encargadas de representar visualmente los datos al usuario. Por último, el controlador es el intermediario entre el modelo y la vista. Recibe los eventos generados por el usuario, modifica el model de acuerdo a los eventos y cuando el modelo no actualiza la vista el controlador se encarga de hacerlo. En la Figura 3.2 se puede observar el comportamiento de este patrón.

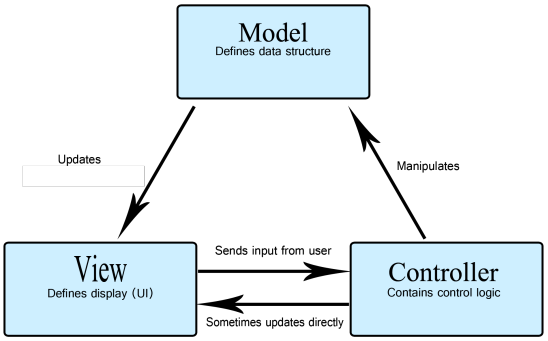

Figura 3.2: Esquema MVC
Fuente: developer.mozilla.org

3.5 Módulos a desarrollar

En este trabajo se explicará la importancia del Bridge ONVIF, es por ello que antes es necesario comentar los módulos que se deben desarrollar para su implementación. En el **Anexo A** se presenta un esquema de los módulos utilizados para este caso de uso.

Los dos primeros módulos que deben ser implementados son los módulos del Servidor ONVIF y Cliente ONVIF, puesto que entre estos dos se debe poder establecer una conexión antes de continuar con los otros módulos. Seguidamente, se implementará el módulo del Servidor Web, que junto con el Cliente ONVIF serán el Bridge ONVIF.

Por último, se utilizará un módulo de una aplicación con interfaz privada, que será la encargada de enviar notificaciones HTTP al Bridge ONVIF. Este será el único propósito de este módulo, no podrá recibir mensajes ni establecer conexiones con otro módulo.

3.5.1. Servidor ONVIF

Para comenzar con el diseño de este módulo es necesario tener claro cuáles serán las funcionalidades y objetivos de este. En el **Anexo A** se presenta un esquema de la implementación del Servidor ONVIF.

En primer lugar, el servidor será el encargado de recibir mensajes por parte del cliente, en concreto, recibirá la url RTSP del dispositivo que se ha seleccionado. El protocolo RTSP *"es un protocolo de flujo de datos en tiempo real no orientado a conexión que se utiliza para definir cómo se hará el envío de información entre el cliente y el servidor"* [21], que en concreto se utilizará para poder visualizar el streaming en vivo del dispositivo. Esta url se almacenará internamente, puesto que se utilizará en otras ocasiones.

En segundo lugar, el servidor recibirá notificaciones de eventos generados por el dispositivo seleccionado. En la sección del módulo del Cliente ONVIF se explicará en detalle el concepto de notificaciones y los eventos. La notificaciones recibidas serán almacenadas, ya que luego serán filtradas comprobando si cumplen las características deseadas.

Por último y tras el filtrado de notificaciones, se procederá a reproducir en vivo el streaming del dispositivo previamente seleccionado por el Cliente ONVIF.

Por otro lado, el servidor ONVIF tendrá otra funcionalidad y es que podrá recibir notificaciones de un sistema externo. El Bridge ONVIF recibirá notificaciones de una aplicación con interfaz privada y será el encargado de reenviárselas al Servidor ONVIF.

3.5.2. Cliente ONVIF

En el **Anexo A** se puede visualizar el esquema utilizado para el diseño de este módulo. Se puede observar que hay dos esquemas, uno cuando se utiliza el cliente para el caso de uso del Bridge ONVIF y otro cuando es utilizado con el Servidor ONVIF. En primer lugar, se detallará el diseño del caso del Servidor ONVIF y por último, el diseño del Bridge ONVIF.

En primer lugar, este módulo debe tener la capacidad de descubrir los dispositivos ONVIF en red. Los dispositivos son descubiertos en la red local por el cliente utilizando el **Web Service Dynamic Discovery (WS-Discovery)**. Este permite que el cliente pueda recibir información detallada sobre el dispositivo, ya que una vez descubierto el dispositivo se conoce su endpoint, que tiene el siguiente formato: *http://onvif_host/onvif/device_service*.

Un dispositivo compatible con ONVIF tiene que facilitar la gestión del mismo y el servicio de eventos. Además, deberá proporcionar el alcance del mismo. Si el dispositivo soporta determinados servicios, deberá ser capaz de responder a todos los comandos de dicho servicio. En caso de que no se soporte uno de los servicios, el dispositivo deberá responder la petición con alguna de las siguientes respuestas: **env:Receiver** o **ter:ActionNotSupported**. [22]

Tras la elección del dispositivo, se obtendrá la url RTSP para poder transmitir el streaming en vivo. Para ello se hace uso del Media WSDL que es el servicio que proporciona el dispositivo para lo relacionado con la configuración de las propiedades de streaming. Es importante disponer del usuario y contraseña para poder visualizar la imagen en vivo del dispositivo, por lo tanto, el cliente debe conocer ambos. El cliente debe descubrir en red al servidor puesto que le enviará la url, ya con el usuario y contraseña, para que él transmita el streaming.

Luego de recibir la respuesta del servidor, se procederá a la suscripción del dispositivo. Cuando se habla de suscripción denota que el cliente podrá suscribirse a un evento detectado por el dispositivo y este enviará notificaciones a una dirección previamente elegida por el cliente. En este proyecto se utilizará el servidor para recibir notificaciones. Un evento es una acción u ocurrencia detectada por un dispositivo. Los eventos se manejan por medio del servicio de evento definido en el ONVIF Event WSDL. Un dispositivo compatible con ONVIF debe proporcionar este servicio. [22]

Las especificaciones de ONVIF establecen tres tipos de notificaciones, pero en este proyecto se gestionarán los eventos mediante la interfaz de notificación básica, WS-BaseNotification. WS-BaseNotification es el mecanismo estándar de notificación WS. Se configura una suscripción, y el servicio que maneja la notificación se conecta a la URL especificada y publica el mensaje de la notificación. La conexión en la que se envía la notificación es iniciada por el productor, y el consumidor no necesita ser la misma entidad que configura la configura la suscripción [23]. En la Figura 3.3 se puede observar el diagrama de secuencia de las notificaciones utilizadas en este proyecto. El *Event Service* y *Subscription Manager* tienen que estar instanciados por el dispositivo.

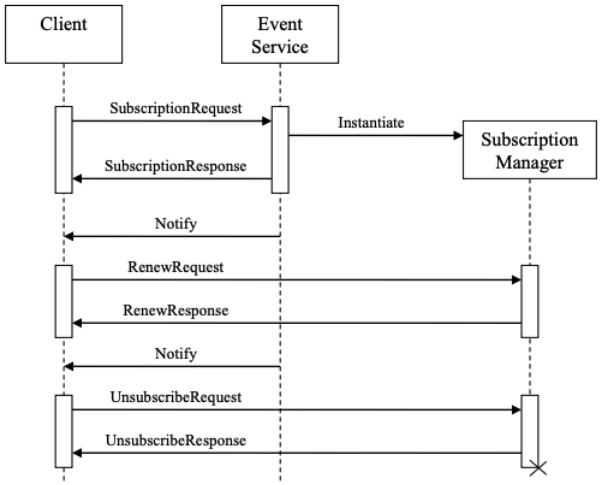

Figura 3.3: Diagrama de secuencia de la interfaz de notificación básica
Fuente: ONVIF Core Specifications - Ver. 19.12

Para poder realizar una suscripción, se debe establecer una conexión entre el cliente y el Event Service. Si la suscripción se ha realizado correctamente y es aceptada, el Event Service instanciará al Subscription Manager que es el que representará la suscripción. En la respuesta, el Event Service devuelve el endpoint del Subscription Manager. El cliente tiene la opción de establecer un tiempo en el que finalizará la suscripción, y al finalizar se destruirá automáticamente el Subscription Manager. Si se desea, se puede posponer la finalización de la suscripción enviando *RenewRequests* al Subscription Manager. Además, el cliente tendrá la opción de finalizar la suscripción antes de que transcurra el tiempo con la petición *UnsubscribeRequest*, que será enviada al Subscription Manager utilizando el endpoint obtenido.

Mediante la conexión entre el cliente y el Event Service, el Event Service envía notificaciones en cualquier momento de la suscripción a la dirección elegida por el cliente. En el **Anexo C** se muestra un ejemplo del contenido de una notificación. Para el envío se utilizan mensajes HTTP POST donde el body del mensaje es la notificación escrita en XML.

Por último, el cliente le informa al servidor que la suscripción ya ha finalizado y este comenzará la transmisión del streaming.

Otra funcionalidad de este módulo, es comportarse como un buscador para el Bridge ONVIF. Será un buscador porque se encargará de descubrir Servidores ONVIF en la red para luego poder enviar la dirección del servidor previamente elegido por el cliente al Bridge.

3.5.3. Servidor Web

Este módulo debe ser capaz de recibir notificaciones HTTP de un sistema externo, en consecuencia, su dirección IP tendrá que ser fija y pública. Además, una conexión debe ser establecida entre el Cliente ONVIF y este módulo, dado que, el cliente ONVIF se comportará como un buscador de servidores ONVIF. Es necesario un buscador, ya que, las direcciones IP de los servidores son privadas y van variando.

En primer lugar, estará recibiendo notificaciones HTTP de una aplicación con interfaz privada. Esto lo estará haciendo continuamente, a su vez, deberá estar a la escucha de una conexión con el cliente ONVIF. El cliente ONVIF, le enviará en un mensaje la dirección IP y puerto donde estará escuchando el servidor ONVIF. Luego, una vez obtenida la dirección del servidor, este módulo reenviará las notificaciones del sistema externo al servidor, permitiendo de esta forma integrar notificaciones HTTP en un servidor ONVIF.

3.6 Integración de módulos

Para la integración de los módulos se utilizarán dos protocolos: HTTP y TCP. Se decidió emplear el protocolo TCP porque asegura que los mensajes entre dos sistemas lleguen a su destino sin errores y en el orden en el que son enviados. Además, permite controlar el flujo, permitiendo moderar la saturación del host remoto. Estos protocolos serán utilizados en las comunicaciones entre:

- **Protoloco TCP:**
  - Cliente ONVIF y Servidor ONVIF: en el envío de la url RTSP del dispositivo.
- **Protocolo HTTP:**
  - Dispositivo y Servidor ONVIF: en el envío de notificaciones a la hora de suscribirse en un evento.
  - Aplicación con interfaz privada y Servidor Web: envío de notificaciones.
  - Servidor Web y Servidor ONVIF: en el reenvío de las notificaciones del sistema externo.
  - Cliente ONVIF y Servidor Web: en el envío de la dirección IP del servidor seleccionado.

4

Desarrollo

4.1 Introducción

Al finalizar la fase de Diseño, se procede al Desarrollo del proyecto. Realizando un buen diseño se consigue reducir el tiempo invertido en esta etapa, que dentro de todas las fases es la más larga.

Los dos primeros módulos que deben ser implementados son los módulos del Servidor ONVIF y Cliente ONVIF, puesto que entre estos dos se debe poder establecer una conexión antes de continuar con los otros módulos. Para implementarlos se utilizará **Qt**. *"Disponible para Linux, Windows, MacOS, Android, iOS, etcétera, no es un lenguaje de programación se trata de un framework escrito en C++"* [24]. Se ha decidido utilizar Qt para el desarrollo de estos módulos dado que permite utilizar una amplia variedad de librerías, es de código abierto por lo que cuenta con el suficiente soporte, entre otras cosas.

Tras la comprobación del correcto funcionamiento de los primeros dos módulos, se implementará el módulo de la aplicación con interfaz pública, es decir el Servidor Web, que junto con el Cliente ONVIF serán el Bridge ONVIF. Este módulo será desarrollado utilizando **Node.js**. *"Es un entorno que trabaja en tiempo de ejecución, de código abierto, multi-plataforma, que permite a los desarrolladores crear toda clase de herramientas de lado servidor y aplicaciones en JavaScript"* [25]. Se decidió utilizar este entorno ya que las peticiones web son tratadas en un mismo hilo, al contrario que los servidores web tradicionales que requieren múltiples hilos para tratar las peticiones. Otra característica por la que se decidió utilizar este entorno fue la facilidad con la que se puede desarrollar un servidor web. [26]

4.2 Servidor y Cliente ONVIF

En primer lugar, se comenzó realizando programas simples utilizando Qt ya que era la primera vez que lo utilizaba. Las pruebas consistieron en seguir tutoriales online de aplicaciones de consola

que es lo que se desarrolló en el proyecto. Una vez que los conceptos básicos fueron consolidados, se continuó realizando pruebas, pero enfocadas a implementar un servidor y un cliente que eran los módulos necesarios para este proyecto. Para ello se consultaron los *Network Examples* de la página oficial de Qt. En concreto, se utilizaron los ejemplos *Fortune Client* y *Fortune Server* [27]. Este ejemplo cuenta con un cliente y un servidor, donde se establece una conexión y el cliente le solicita al servidor frases y el servidor se las envía eligiendo aleatoriamente de un listado. Si bien el ejemplo se trata de una aplicación con *widgets*, resultó útil ya que se puede ver cómo se comunican los elementos. Además, cada ejemplo en la página cuenta con documentación donde se detalla todo lo utilizado.

Luego de absorber los conocimientos aprendidos en el tutorial y en los ejemplos, se desarrolló un programa básico donde un cliente y servidor se comunicarán y se enviarán mensajes. Para ello se crearon dos proyectos en Qt Creator, uno para cada módulo. Al principio lo que me resultó complicado fue conseguir leer la información que se estaban enviando, ya que los ejemplos que en los que me estaba basando usaban distintos métodos, pero una vez conseguido pude continuar. Es importante destacar que en primer lugar, se debe ejecutar el servidor ya que quedará a la escucha de cualquier conexión. Luego se ejecuta el cliente y se establecerá la conexión.

Para el desarrollo de estos módulos, se utilizó el código obtenido de un repositorio de *Github* [28], que cuenta tanto con una librería ONVIF como con un cliente y servidor. Fue necesario adaptarlo ya que el código estaba implementado para ser ejecutado en un sistema operativo distinto a MacOS, por lo que se tuvieron que añadir la siguientes líneas de código en algunos archivos de la librería:

```
#if !defined(SOL_TCP) && defined(IPPROTO_TCP)
#define SOL_TCP IPPROTO_TCP
#endif
#if !defined(TCP_KEEPIDLE) && defined(TCP_KEEPALIVE)
#define TCP_KEEPIDLE TCP_KEEPALIVE
#endif
```

Código 4.1: Solución del problema en la librería ONVIF

En este repositorio se encontraba una aplicación de *widgets*, por lo que se tuvo que adaptar para que sea una aplicación de desarrollo. Los primeros cambios realizados fueron en el servidor ya que la dificultad no era alta. Tras la comprobación de que no se vio afectada ninguna funcionalidad, se modificó el cliente que es donde se debían realizar más cambios. Una vez que se comprobó que ambos módulos funcionaban correctamente, se continuó con el desarrollo del proyecto, el servidor debe ser capaz de reproducir la imagen en vivo del dispositivo seleccionado. Se contemplaron distintas herramientas para la reproducción como *QMediaPlayer, VLC library* y *OpenCV*. Al final, se decidió utilizar OpenCV por estar familiarizada con ella tras haberla utilizado en la asignatura **Redes de Comunicaciones 2**.

Para conseguir que el servidor reproduzca la imagen en vivo del dispositivo, es necesario que se establezca una conexión entre el cliente y el servidor, puesto que será el cliente quien le enviará al servidor la url RTSP del dispositivo previamente descubierto en red y seleccionado. Cabe destacar que se debe conocer el usuario y la contraseña del dispositivo seleccionado, ya que es importante que la url disponga de ambos datos. Si no los dispone, no se podrá visualizar la imagen. El envío de la url se realiza mediante el protocolo TCP, ya que es importante que no se pierda ninguna información y que los mensajes lleguen en el orden en los que fueron enviados.

La reproducción fue conseguida utilizando el método que se encuentra en el **Anexo B**. Fue

necesario modificar este método ya que OpenCV utiliza TCP como protocolo de reproducción y se utilizará UDP ya que es utilizado por RTSP para los datos de vídeo y de audio. Además de reproducir la imagen en vivo, se almacenará en un archivo en el destino que se desee. Para utilizar OpenCV con el protocolo UDP fue necesario modificar el *environment*.

```
setenv("OPENCV_FFMPEG_CAPTURE_OPTIONS", "rtsp_transport;udp", 1);
```

Código 4.2: Modificar protocolo OpenCV

Una vez que se obtuvo la reproducción de la imagen en vivo, se continuó con el desarrollo del cliente, es decir, el cliente además de enviarle la url de streaming al servidor debe ser capaz de poder suscribirse a los eventos generados por un dispositivo. Para ello fue necesario leer mucha documentación, sobre todo la encontrada en *ONVIF Core Specifications* y en la *ONVIF Application Programmers Guide* para comprender el concepto de evento y entender cómo funcionaba el método *suscribe* del event service. Luego de familiarizarme con los conceptos, se desarrollo la notificación siguiendo el WS-BaseNotification descrito en la fase de Diseño. Para ello fue necesario tener claro a quién se le iban a mandar las notificaciones generadas por el dispositivo. Puesto que el servidor es quien reproducirá la imagen del dispositivo, se seleccionó el servidor como receptor de las notificaciones ya que cuando reciba una notificación del dispositivo se comenzará a reproducir la imagen. Para que la suscripción a los eventos fuese exitosa, es necesario enviar en la petición SOAP la dirección a donde se envían las notificaciones y la duración de la suscripción. En este proyecto, la suscripciones durarán un minuto por cuestiones de pruebas.

El servidor estará a la escucha en un determinado puerto para recibir notificaciones y durante la duración de la suscripción estará recibiendo cualquier notificación que será almacenada en un map para poder luego filtrarlas. Una vez que finaliza la suscripción, el servidor filtrará las notificaciones por la dirección del dispositivo que se ha seleccionado. El servidor conoce la dirección ya que al comienzo de la ejecución el cliente le envía la url RTSP del dispositivo, que contiene la dirección IP del dispositivo. En el **Anexo C** se encuentra el contenido de una notificación, se puede observar que uno de los campos del XML es el *ProducerReference* que es el dispositivo seleccionado.

Tras la finalización de la suscripción el cliente se desconecta y el servidor comienza a reproducir la imagen en vivo. Para finalizar la reproducción se podrá pulsar cualquier tecla del teclado y el servidor se desconectará.

4.3 Bridge ONVIF

Para el desarrollo del Bridge ONVIF fue necesario utilizar el servidor y cliente ONVIF además de un sistema externo que enviará notificaciones HTTP. Fue necesario desarrollar una aplicación con interfaz pública que fuese capaz de recibir y reenviar dichas notificaciones, en concreto se desarrolló un servidor web. En el **Anexo B** se puede observar el código implementado para el servidor web. La suma del cliente ONVIF y el servidor web será el Bridge ONVIF. Es importante destacar que la comunicación entre el cliente ONVIF y el servidor web será mediante peticiones HTTP POST.

En primer lugar se ha desarrollado el servidor web y se ha modificado el cliente ONVIF. Este fue el primer paso puesto que entre ambos debe existir una conexión donde se enviará la dirección del servidor ONVIF. El servidor web tiene que ser capaz de poder identificar el

mensaje del cliente dentro de las notificaciones que recibe. Es por ello que el mensaje que le enviará el cliente comenzara con las palabra *server* y el servidor web podrá hacer un filtro de las notificaciones. Además, una vez que se ha recibido el mensaje podrá comenzar el reenvío de notificaciones al servidor ONVIF.

Luego de que se reciba el mensaje del cliente, se reenviarán las notificaciones HTTP al servidor ONVIF. Para ello fue inevitable modificar el servidor web puesto que hasta el momento solo recibía notificaciones. El reenvío se hará mediante peticiones POST. El servidor ONVIF también fue necesario modificarlo, para que al comienzo de su ejecución este a la escucha de notificaciones enviadas por el Bridge y no por el cliente ONVIF.

4.4 Herramientas utilizadas

En esta sección se describirán las herramientas que fueron utilizadas para el desarrollo del proyecto.

4.4.1. Servidor y Cliente ONVIF

En el desarrollo de estos módulos se ha utilizado `Qt Creator` tanto como entorno de trabajo como entorno de desarrollo. Además, fue necesario utilizar una máquina virtual con Windows, para poder utilizar `Onvif Device Manager` como administrador de los dispositivos empleados.

4.4.1.1. Qt Creator

Qt Creator [29] es un entorno de desarrollo integrado (IDE) creado a fines de 2009, tras el lanzamiento de Qt 4.5, que puede ser utilizado en los sistemas operativos Linux, MacOS y Windows. Permite a los desarrolladores crear aplicaciones para múltiples plataformas, como aplicaciones móviles Android e iOS, aplicaciones de escritorio y dispositivos integrados. También permite desarrollar aplicaciones que pueden ser desplegadas en navegadores web. Este entorno de desarrollo entiende los lenguajes Qt, C++ y QML.

Figura 4.1: Qt Creator.
Fuente: qt.io

Esta herramienta se ha utilizado como entorno de trabajo y para desarrollar todos los módulos para este caso.

En el **Anexo B** se pueden ver algunas funciones relevantes para el desarrollo de este proyecto utilizando esta herramienta.

4.4.1.2. Onvif Device Manager

Onvif Device Manager [30] es una aplicación gratuita que se utiliza para la administración de servidores de vídeo y cámaras dentro de los sistemas CCTV. Está disponible solo para el sistema operativo Windows.

Figura 4.2: ONVIF Device Manager.
Fuente: onvif.org

Con esta aplicación es posible conocer los datos del dispositivo como el modelo, el fabricante, su dirección IP, entre otras cosas. Además, se puede tanto visualizar el streaming en vivo de un cámara, por ejemplo, como obtener la url RTSP para su transmisión del dispositivo seleccionado.

Se ha utilizado para poder administrar los distintos dispositivos que se utilizaron para realizar pruebas en este caso de uso.

4.4.1.3. Virtual Box

Virtual Box [31] es un software de visualización multiplataforma que permite extender la computadora existente para ejecutar múltiples sistemas operativos al mismo tiempo. Está disponible para los sistemas operativos Linux, MacOS, Windows y Oracle Solaris. Con esta herramienta el usuario tiene la posibilidad de crear máquinas virtuales con soporte Linux, Macintosh, Windows, Oracle Solaris, Open Solaris y OpenBSD.

Figura 4.3: Virtual Box logo.
Fuente: virtualbox.org

Esta herramienta se ha utilizado para crear una máquina virtual con el sistema operativo Windows para poder, de esta forma, utilizar el entorno de administración Onvif Device Manager.

4.4.1.4. OpenCV

OpenCV [32] es una biblioteca escrita en C y C++ de código abierto disponible para Linux, Windows y Mac OS X. Uno de los objetivos de OpenCV es permitir a los usuarios que lo utilicen crear aplicaciones visuales. Esto lo consiguen facilitando la infraestructura fácil de usar de visión por computadora.

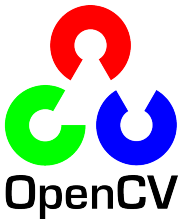

Figura 4.4: OpenCV logo.
Fuente: opencv.org

Esta herramienta ha sido utilizada para reproducir el streaming en vivo del dispositivo seleccionado. En el **Anexo B** se puede ver un ejemplo de cómo se ha empleado esta herramienta.

4.4.2. Bridge ONVIF

Para el desarrollo de este caso, se han utilizado las mismas herramientas que para el Servidor y Cliente ONVIF. Además, se ha utilizado Atom como entorno de desarrollo. Cabe destacar que para este caso se ha utilizado una máquina virtual con el sistema operativo Linux donde se ha desarrollado el Bridge.

4.4.2.1. Atom

Atom [33] es un editor de código gratuito disponible para Linux, MacOS y Windows. Es desarrollado por Github, permitiendo utilizar Git para el control de versión y la instalación de múltiples *plug-ins*. Soporta múltiples lenguajes como Java, Python, C++, HTML, JavaScript, etc. En este caso, se va a utilizar para desarrollar en JavaScript.

Figura 4.5: Atom.
Fuente: atom.io

Se ha elegido este editor de código para desarrollar el Servidor Web porque permite incorporar librerías que comprueban la sintaxis del código, por ejemplo, permitiendo así no tener errores de compilación.

En el **Anexo B** se puede visualizar el código empleado para crear el módulo del servidor web que es un elemento del Bridge ONVIF.

4.4.3. Documentación

Me gustaría comentar que se ha utilizado la herramienta *Latex* para el desarrollo de este documento. Durante la carrera no había tenido la oportunidad de utilizarla y puesto que permite crear documentos con un estilo marcado y fácilmente estructurado además de una creación de manera simple, creí que era un buen momento para utilizarla.

5

Integración, pruebas y resultados

5.1 Introducción

Durante la etapa de Desarrollo se realizaron pruebas para comprobar el comportamiento de la aplicación. En este capítulo, se expondrán dichas pruebas y se describirán sus resultados. Además, se expondrán las pruebas realizadas luego de la fase de Desarrollo.

En primer lugar, se comenzó realizando pruebas de Qt ya que no había utilizado nunca el *framework*. Fue necesario realizar pequeños tutoriales y luego se ejecutaron ejemplos de proyectos con servidores y clientes encontrados en la sección *Network Examples* de Qt. Luego se comentarán las pruebas de los módulos desarrollados que fue donde más pruebas fue necesario realizar.

Por último, se realizan las pruebas de validación que es donde se comprueba que los requisitos analizados en el Diseño hayan sido cumplidos. Es por ello que se ha ejecutado la aplicación revisando los requisitos por subsistemas para comprobar que fueron considerados.

5.2 Pruebas a lo largo del ciclo de vida

5.2.1. Pruebas de Caja Negra

En estas pruebas no se tiene en cuenta el comportamiento interno de la aplicación, sólo se tienen en cuenta los datos de entrada y salida.

Para realizar pruebas del Bridge ONVIF fue necesario crear un pequeño programa que actuará como el sistema externo. Este programa sólo enviará peticiones HTTP POST con una dato aleatorio par. Para obtener un resultado exitoso es necesario que el servidor web sea el primero en ser ejecutado. Una vez que se comprueba que el servidor web está activo, se debe ejecutar el sistema externo que será quién le envíe las notificaciones HTTP al servidor web. Antes de que el cliente ONVIF se conecte y establezca una conexión con el servidor web, el servidor ONVIF

se debe ejecutar. En el caso de que se generé un error durante la ejecución, se imprimirá por pantalla dicho error registrando donde ocurrió.

Las primeras pruebas realizadas en el caso del Servidor ONVIF no fueron exitosas por el sistema operativo que se utilizó para el desarrollo. Tras realizar los cambios correspondientes se volvió a realizar una prueba donde se comprueban todas las funcionalidades de tanto el servidor como el cliente. Para realizar las pruebas se utilizará una cámara IP.

Antes de comenzar con la visualización en OpenCV, se realizaron varias pruebas conectando la cámara y utilizando la herramienta Onvif Device Manager. Desde esta, se obtuvo la url RTSP que es necesaria para visualizar la imagen con OpenCV.

Una vez modificado el protocolo, se realizaron pruebas introduciendo la url RTSP directamente en el código. Al principio el método implementado no funcionaba correctamente y no se podía ”abrir” la cámara. Esto se debía a que es necesario conocer el usuario y contraseña de la cámara para poder visualizar su imagen. Luego de introducirlos se pudo ver el streaming correctamente, por lo que el siguiente paso sería conseguir la url dinámicamente sin utilizar una herramienta externa.

La siguiente prueba realizada, una vez visualizada correctamente la cámara obteniendo la url de forma dinámica, fue conseguir una suscripción a un evento exitosa. Para esta prueba fue necesario leer mucha documentación, sobre todo la encontrada en *ONVIF Core Specifications* y en la *ONVIF Application Programmers Guide* para comprender el concepto de evento y entender cómo funcionaba el método *suscribe* del event service.

Se han realizado pruebas ejecutando primero el cliente y luego el servidor, no afecta hasta el momento en que el cliente le debe enviar la url al servidor, si no está a la escucha de mensajes no se podrá visualizar la imagen del dispositivo seleccionado. Otra prueba que ha fracasado, es tirar el servidor en mitad de la ejecución, ya que el servidor va a estar esperando la url en un puerto y se habrá establecido la dirección del servidor en un puerto diferente para la recepción de notificaciones.

La última prueba realizada para comprobar que el cliente y servidor ONVIF funcionaban correctamente, fue seguir el esquema de ambos y comprobar que tras la recepción de las notificaciones el servidor mostraba por pantalla la imagen de la cámara.

El caso del Servidor ONVIF tiene que seguir los siguientes pasos para ser ejecutado correctamente:

1. Ejecutar el servidor ONVIF.

2. Ejecutar el cliente ONVIF que será quién descubra los dispositivos ONVIF en la red. Si hay más de un dispositivo, el usuario tendrá la opción de elegir. En caso contrario se selecciona automáticamente el único dispositivo en red.

3. El usuario deberá seleccionar la url RTSP para la reproducción. Siempre que haya más de una url, el usuario tendrá la opción de elegir, en caso contrario la selección se hace automáticamente.

4. Luego de la elección de la url se debe introducir el usuario y la contraseña del dispositivo seleccionado. Se enviará al servidor la url ya compuesta, es decir, *http://user:password@host:port/*. El servidor mostrará por pantalla la url que recibe.

5. En cuanto el servidor recibe la url, el cliente inicia la suscripción a los eventos del dispositivo. Mientras tanto, el servidor quedará a la espera de recibir notificaciones ya que el cliente le informará al dispositivo que debe enviárselas a él.

6. El cliente es el encargado de informarle al servidor que la suscripción ha finalizado, su duración es de un minuto. El servidor en cuanto recibe el mensaje del cliente comienza a filtrar las notificaciones recibidas, que fueron almacenadas en un listado.

7. Si una de las notificaciones recibidas cumple los requisitos, ser del dispositivo seleccionado y que se haya generado un evento, se comienza a reproducir la imagen en vivo del dispositivo seleccionado. En caso contrario, la notificación será eliminada.

8. La reproducción finaliza en cuanto se pulse cualquier tecla. La reproducción quedará almacenada en el destino que se haya seleccionado.

9. El servidor se desconectará tras este suceso junto con el cliente.

5.2.2. Pruebas de Caja Blanca

En estas pruebas es necesario conocer la lógica interna del programa ya que será utilizada para examinar el comportamiento y la estructura de la aplicación.

Para realizar estas pruebas se utilizó la herramienta **Qt Test** que permite realizar pruebas unitarias basadas en Qt. De esta forma fue posible probar todos los módulos y su comportamiento. Si un módulo pasa una prueba se continuaba con el siguiente. En caso de que surgiera un error se solucionaba y hasta que no se no quedara resuelto no se pasaba al siguiente módulo.

6

Conclusiones y trabajo futuro

En este Trabajo de Fin de Grado se consiguió crear un nuevo dispositivo, un Bridge ONVIF, que lo pueden utilizar aquellos que cuenten con uno o varios sistemas ONVIF y deseen integrar HTTP en ellos. El Bridge ONVIF permite la comunicación mediante HTTP entre dos dispositivos, donde uno de ellos es compatible con el estándar ONVIF.

Algunos de los dispositivos que utilizan el estándar fueron detallados en este documento. Gracias al Bridge ONVIF se consiguió la comunicación entre dos sistemas, donde uno de ellos debe contar con algún dispositivo, como los descritos, compatible con el protocolo. Esto es una ventaja puesto que un sistema que cuenta con dispositivos IoT o aplicaciones inteligente, por ejemplo, podrá integrarse mediante HTTP en un sistema ONVIF.

No se ha implementado todo el estándar, sólo se ha implementado el descubrimiento de dispositivos ONVIF en red, el envío de la url de streaming para la reproducción y por último, se implementó la suscripción a los eventos de un dispositivo. Con esto se logró crear un Servidor ONVIF capaz de reproducir la imagen en vivo de un dispositivo previamente seleccionado. Esto permite, por ejemplo si el dispositivo en cuestión es una cámara de red, comenzar a grabar en cuanto se genere un evento que puede ser el evento de detección de movimiento. De esta forma, se reduciría el espacio ocupado en el almacenamiento del Servidor ONVIF.

El Servidor ONVIF es quien recibe las notificaciones HTTP reenviadas por el Bridge ONVIF y además, las generadas por el dispositivo durante la suscripción a un evento. El Servidor fue desarrollado de tal forma que estará escuchando cualquier mensaje que sea enviado, incluyendo las notificaciones que luego serán filtradas por la dirección IP del dispositivo seleccionado. El Servidor, en un trabajo a futuro, podría ser implementado de forma que no tenga que hacer un filtro de los mensajes recibidos, sino que solo pueda recibir las notificaciones reenviadas por el Bridge o las generadas por el dispositivo.

Como se comentó en la fase de Desarrollo, se ha utilizado Qt para desarrollar la mayor parte de este proyecto. La otra parte, que sería la implementación del Servidor Web, fue desarrollada utilizando JavaScript. Si tuviera que volver a realizar el proyecto optaría por utilizar un único lenguaje y sería Qt. Qt utiliza el lenguaje de programación C++ de forma nativa y en los años

cursados nunca lo había utilizado. Si bien había utilizado C en algunas asignaturas y C++ incluye este, al comienzo no estaba muy entusiasmada en usarlo, pero tras leer documentación y realizar tutoriales comprobé que era un lenguaje sencillo de utilizar y que había demasiada información y ejemplos donde se empleaba Qt para aplicaciones ONVIF. En la página oficial de ONVIF, hay ejemplos donde uno de ellos utiliza Qt para el desarrollo de una aplicación Android. Por lo tanto, teniendo en cuenta todo lo expuesto creo que fue un acierto utilizar Qt para el desarrollo del proyecto.

En un futuro se podría trabajar en implementar un **sistema externo** desarrollando una API que permita el envío de notificaciones HTTP, con o sin información, a una dirección pública determinada. Por otro lado, se podría implementar una **cámara virtual** que disponga de un servidor de vídeo y un cliente ONVIF. Esta cámara permitiría mandar un vídeo a cualquier dispositivo ONVIF. Además, sería especialmente útil para probar soluciones de analíticas de vídeo ya que permite automatizar las pruebas.

Por último, me gustaría destacar que el trabajo realizado en este proyecto permitió crear, por un lado, un nuevo dispositivo que podrá ser utilizado por un amplio número de empresas que busquen la integración de dos sistemas empleando HTTP como protocolo de comunicación. Por otro lado, fue implementado parte del protocolo ONVIF que permite la reproducción del streaming de un dispositivo a partir de la suscripción de un evento.

Bibliografía

[1] M. D. P. A. RAMOS and A. G.-C. HURTADO, *Seguridad Informatica ED. 11 Paraninfo*. Editorial Paraninfo, 2011.

[2] Our Mission. [Online]. Available: https://www.onvif.org/about/mission/

[3] "Niveles de membresía." [Online]. Available: https://www.onvif.org/join-us/membership-levels/

[4] "Understand onvif in 10 minutes." [Online]. Available: https://www.onvif.org/wp-content/uploads/2020/04/Understand-ONVIF-in-10-Minutes.pdf

[5] "Onvif profiles." [Online]. Available: https://www.onvif.org/profiles/

[6] "Conformant products." [Online]. Available: https://www.onvif.org/conformant-products/

[7] F. J. G. Mata, *Videovigilancia: CCTV usando vídeos IP*. Editorial Vértice, 2010.

[8] J. Cabasso, "Analog vs. ip cameras," *Aventura Technologies Newsletter*, vol. 1, no. 2, pp. 1–8, 2009.

[9] J. Church, D. Martz, and N. Cook, "The use of digital video recorders (dvrs) for capturing digital video files for use in both the observer and ethovision," *Behavior research methods*, vol. 38, no. 3, pp. 434–438, 2006.

[10] "Digital video recorder (dvr)." [Online]. Available: https://www.techopedia.com/definition/4702/digital-video-recorder-dvr

[11] F. Nilsson *et al.*, *Intelligent network video: Understanding modern video surveillance systems*. CRC Press, 2016.

[12] O. Vellacott, "¿ por qué usar video ip?" 2014.

[13] D. Neal and S. Rahman, "Video surveillance in the cloud?" *arXiv preprint arXiv:1512.00070*, 2015.

[14] J. Barendt and K. Fransson, "Generic event integration in video management software," *LU-CS-EX 2018-37*, 2018.

[15] "The advantages of video management software." [Online]. Available: http://www.cyberinfrared.com/the-advantages-of-video-management-software/

[16] "Web service." [Online]. Available: https://techterms.com/definition/web_service

[17] O. Vellacott, "La apertura de las soluciones de seguridad por ip."

[18] "Onvif benefits for end users." [Online]. Available: https://www.onvif.org/benefits/end-users

[19] "Onvif benefits for system integrators/specifiers." [Online]. Available: https://www.onvif.org/benefits/onvif-benefits-for-system-integratorsspecifiers/

[20] G. Arcos-Medina, J. Menéndez, and J. Vallejo, "Comparative study of performance and productivity of mvc and mvvm design patterns," *KnE Engineering*, pp. 241–252, 2018.

[21] D. M. Costilla and S. R. Montoro, "Streaming de audio/video. protocolo rtsp," *Serveis Telemàtics*, pp. 2–3, 2008.

[22] "Onvif core specifications ver 19.12." [Online]. Available: https://www.onvif.org/specs/core/ONVIF-Core-Specification.pdf

[23] "Onvif application programmers guide." [Online]. Available: https://www.onvif.org/wp-content/uploads/2016/12/ONVIF_WG-APG-Application_Programmers_Guide-1.pdf

[24] "About qt." [Online]. Available: https://wiki.qt.io/About_Qt/es

[25] "Introducción a express/node." [Online]. Available: https://developer.mozilla.org/es/docs/Learn/Server-side/Express_Nodejs/Introduction

[26] B. Dayley, *Node. js, MongoDB, and AngularJS web development.* Addison-Wesley Professional, 2014.

[27] "Network examples." [Online]. Available: https://doc.qt.io/qt-5/examples-network.html

[28] S. Hummatli, "Onvif qt server and client," https://github.com/hummatli/onvif-qt-server-client, 2016.

[29] "Qt creator manual." [Online]. Available: https://doc.qt.io/qtcreator/index.html

[30] Synesis, *ONVIF Device Manager, User Guide*, Synesis.

[31] "Virtual box." [Online]. Available: https://www.virtualbox.org/

[32] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library.* .º'Reilly Media, Inc.", 2008.

[33] "Atom." [Online]. Available: https://atom.io/

Acrónimos

HTTP Hypertext Transfer Protocol.

HTTPS HyperText Transport Protocol Secure.

IDE Entorno de Desarrollo Integrado.

IoT Internet of Things.

IP Internet Protocol.

NVR Grabador de Vídeo de Red.

ONVIF Open Network Video Interface Forum.

PTZ Pan Tilt Zoom.

RTSP Protocolo de Transmisión en Tiempo Real.

SD Secure Digital.

SOAP Simple Object Access Protocol.

TCP Transmission Control Protocol.

TFG Trabajo de Fin de Grado.

TLS Seguridad de la Capa de Transporte.

VMS Video Management System.

WS Servicio Web.

WS-Discovery Web Service Dynamic Discovery.

WS-I Organización para la Interoperabilidad de Servicios Web.

WSDL Web Services Description Language.

XML Extensible Markup Language.

Anexos

A

Esquemas

A.1 Bridge ONVIF

En este apartado se muestra el esquema utilizado para diseñar el Bridge ONVIF implementado en este proyecto.

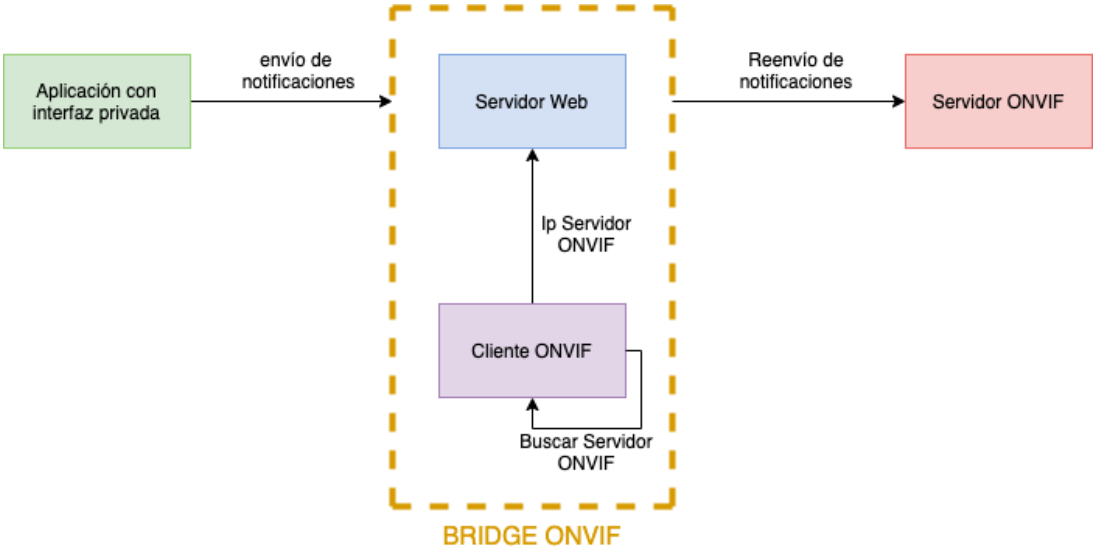

Figura A.1: Esquema Bridge ONVIF.

A.2 Servidor ONVIF

En este apartado se muestra el esquema utilizado para diseñar el Servidor ONVIF, implementado en este proyecto.

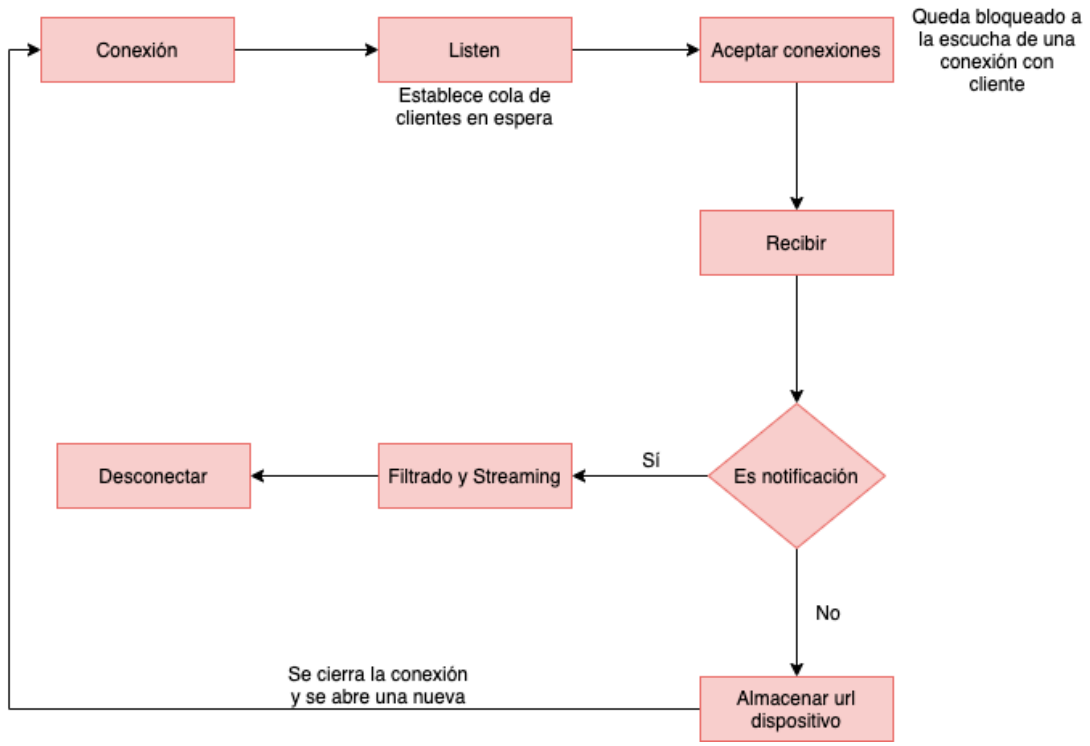

Figura A.2: Esquema Servidor ONVIF.

A.3 Cliente Onvif

En este apartado se muestra en primer lugar el esquema del Cliente ONVIF para el caso de uso del servidor ONVIF y en segundo lugar, el esquema utilizado para el Bridge ONVIF.

A.3.1. Esquema Cliente ONVIF utilizado en el Servidor ONVIF

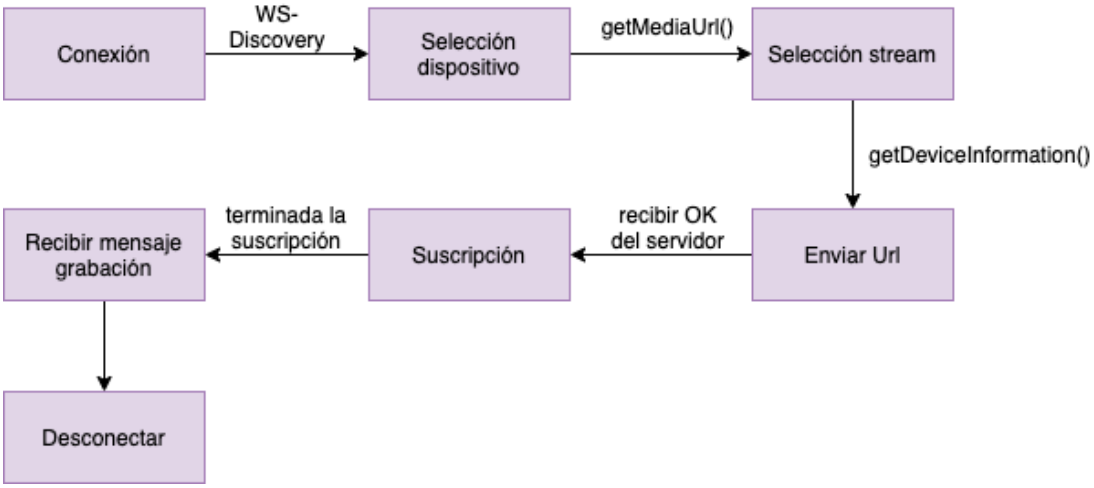

Figura A.3: Esquema Cliente ONVIF utilizado en el Servidor ONVIF.

A.3.2. Esquema Cliente ONVIF utilizado en el Bridge ONVIF

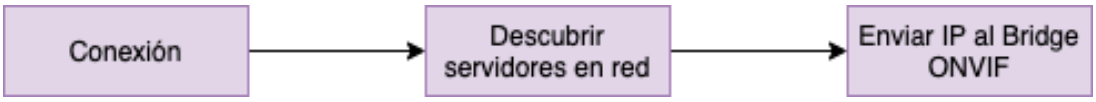

Figura A.4: Esquema Cliente ONVIF utilizado en el Bridge ONVIF.

B

Código

B.1 Url Streaming

En este ejemplo se muestra el método utilizado para la obtención de la URL del dispositivo seleccionado. Previamente a llamar a este método, es necesario comprobar si el dispositivo tiene distintos perfiles, es decir, si cuenta con perfiles de distintas calidades. La variable *profileToken* hace referencia a ese perfil.

```
std::string Media::getStreamURL(QString devServiceURL, std::↩
   string profileToken) {

    std::string uri;

    MediaBindingProxy p;

    _mediaws__GetStreamUri in;
    _mediaws__GetStreamUriResponse out;

    in.ProfileToken = profileToken;

    if (p.GetStreamUri(devServiceURL.toStdString().data(), ↩
       nullptr, &in, out) == SOAP_OK) {
        //ok
        tt__MediaUri * mediaUri = out.MediaUri;
        uri = mediaUri->Uri;

    } else {
        //error
        p.soap_print_fault(stderr);
    }
```

```
    return uri;
}
```

Código B.1: Obtención de la URL de streaming

B.2 OpenCV

En el siguiente ejemplo se muestra por pantalla el código para utilizar la herramienta OpenCV. El streaming se almacena en un fichero a la vez que se muestra por pantalla.

```
void Server::showStream(int notificationNumber){
    int frame_width = 0, frame_height = 0;
    QString path = "path";
    QString videoName = path + "notification" + QString::number(↩
       notificationNumber) + ".avi";
    QString title = "Notification␣" + QString::number(↩
       notificationNumber);

    setenv("OPENCV_FFMPEG_CAPTURE_OPTIONS", "rtsp_transport;udp",↩
        1);
    cv::VideoCapture camera(this->stream.toStdString());


    if (camera.isOpened()){
        qDebug() << "Recording..␣\n(Press␣any␣key␣to␣stop␣↩
           recording)";
        sendMessage();
        frame_width = camera.get(CV_CAP_PROP_FRAME_WIDTH);
        frame_height = camera.get(CV_CAP_PROP_FRAME_HEIGHT);
        VideoWriter video(videoName.toStdString(), CV_FOURCC('M',↩
           'J','P','G'),10, Size(frame_width,frame_height));
        while(1){
            startWindowThread();
            Mat image;
            camera >> image;
            if (image.empty()){
                qDebug() << "Empty␣imaged";
                break;
            }
            video.write(image);
            imshow(getDeviceIP().toStdString(), image);
            int key = waitKey(1);
            if (key > 0 && key < 127){
                destroyWindow(getDeviceIP().toStdString());
                waitKey(1);
                qDebug() << "Video␣taped";
                break;
            }
```

```
        }

        camera.release();
        video.release();

        disconnect();


    }else{
        qDebug() << "Camara␣closed";
    }

}
```

Código B.2: Streaming con OpenCV

B.3 Aplicación con interfaz pública

En este segmento de código se muestra la implementación del servidor web, que será utilizada en el Bridge ONVIF.

```
const express = require('express');
const server = express();
const request = require('request');
const port = 6007;
var flag = "";
var bodySend = "";
var ip = "";


function anyBodyParse(req, res, next){
  var data = '';
␣␣req.rawbody␣=␣'';
  req.setEncoding('utf8');
  req.on('data', function(chunk){
    req.rawbody += chunk;

  });
  req.on('end', function(){
    next();
  });
}

server.use(anyBodyParse);

server.post('/', function(req, res){
  console.dir(req.rawbody)
  if (flag == 1){
```

```
    request.post({
      url: ip,
      body:req.rawbody,
      json:true
    }, function(error, response, body){
      if (!error && response.statusCode == 200){
        console.log(body)
      }
    });
  }else{
    bodySend = req.rawbody
    if(bodySend.includes("server")){
      flag=1
      ip = bodySend.split("=")[1]
    }
  }

  res.contentType('application/xml');
  res.send(req.body, 200);
});

server.get('/', (req, res) => {
  res.sendFile(__dirname + '/index.html');
});

server.listen(port, () => {
  console.log('Server listening at %d', port);
})
```

Código B.3: Servidor Web

C

Suscripción

C.1 Suscripción

En este ejemplo, se observa cómo se realiza una suscripción a un dispositivo permitiendo enviar notificaciones cuando un evento es generado.

```
int Event::getSuscription(QString devServiceURL, QString ↩
    consumerReference) {

    PullPointSubscriptionBindingProxy d;

    _wsnt__Subscribe in;
    _wsnt__SubscribeResponse out;

    std::string p = "PT60S"; // duracion de la suscripcion
    char *aux;

    in.InitialTerminationTime = &p;
    aux = strcpy(new char[consumerReference.toStdString().length↩
        () + 1], consumerReference.toStdString().c_str());
    in.ConsumerReference.Address = aux; //direccion donde se ↩
        envian las notificaciones del evento generado

    if (d.Subscribe(devServiceURL.toStdString().c_str(), nullptr,↩
        &in, out) == SOAP_OK) {
        //ok
        qDebug() << "\tSubscription␣Address:␣" << out.↩
            SubscriptionReference.Address;
        return 1;
```

```
    } else {
        //error
        d.soap_print_fault(stderr);
        return -1;
    }

}
```

Código C.1: Suscripción a un dispositivo

C.2 Notificación

En esta sección se muestra un ejemplo de la notificación enviada por el Event Service durante la suscripción.

```
<?xml version="1.0" encoding="UTF-8"?>
<s:Envelope
    xmlns:s="http://www.w3.org/2003/05/soap-envelope"
    xmlns:e="http://www.w3.org/2003/05/soap-encoding"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:wsa="http://www.w3.org/2005/08/addressing"
    xmlns:xmime="http://www.w3.org/2005/05/xmlmime"
    xmlns:tns1="http://www.onvif.org/ver10/topics"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:xop="http://www.w3.org/2004/08/xop/include"
    xmlns:tt="http://www.onvif.org/ver10/schema"
    xmlns:wsnt="http://docs.oasis-open.org/wsn/b-2"
    xmlns:wstop="http://docs.oasis-open.org/wsn/t-1"
    xmlns:tds="http://www.onvif.org/ver10/device/wsdl"
    xmlns:trt="http://www.onvif.org/ver10/media/wsdl"
    xmlns:tev="http://www.onvif.org/ver10/events/wsdl"
    xmlns:tptz="http://www.onvif.org/ver20/ptz/wsdl"
    xmlns:timg="http://www.onvif.org/ver20/imaging/wsdl"
    xmlns:ter="http://www.onvif.org/ver10/error" >
    <s:Header>
        <wsa:Action>http://docs.oasis-open.org/wsn/bw-2/↩
           NotificationConsumer/Notify</wsa:Action>
    </s:Header>
    <s:Body>
        <wsnt:Notify>
            <wsnt:NotificationMessage>
                <wsnt:SubscriptionReference>
                    <wsa:Address>http://192.168.0.29:3007/</↩
                       wsa:Address>
                </wsnt:SubscriptionReference>
                <wsnt:Topic Dialect="http://www.onvif.org/ver10/↩
                   tev/topicExpression/ConcreteSet">↩
```

```
                    tns1:RuleEngine/CellMotionDetector/Motion</↩
                    wsnt:Topic>
                <wsnt:ProducerReference>
                    <wsa:Address>http://192.168.0.10:10080/↩
                        event_service/0</wsa:Address>
                </wsnt:ProducerReference>
                <wsnt:Message>
                    <tt:Message UtcTime="2020-04-05T12:26:08Z" ↩
                        PropertyOperation="Changed">
                        <tt:Source>
                            <tt:SimpleItem Name="↩
                                VideoSourceConfigurationToken" ↩
                                Value="11"/>
                            <tt:SimpleItem Name="↩
                                VideoAnalyticsConfigurationToken" ↩
                                Value="12"/>
                            <tt:SimpleItem Name="Rule" Value="↩
                                TestRule"/>
                        </tt:Source>
                        <tt:Data>
                            <tt:SimpleItem Name="IsMotion" Value=↩
                                "true"/>
                        </tt:Data>
                    </tt:Message>
                </wsnt:Message>
            </wsnt:NotificationMessage>
        </wsnt:Notify>
    </s:Body>
</s:Envelope>
```

Código C.2: Notificación de una suscripción